CSSE 374 – Software Design: Project Milestone 3

Objective

In this team assignment, your team is to complete the final of the three Milestones that bridges from the Software Architecture completed in Milestone 1, to the beginnings of the Program-Level design with some initial Design Patterns (Strategy and Observer), and now for Milestone 3 to explore some more sophisticated Design Patterns such as decorator, Factory Method, Abstract Factory, and Command. Note: like Milestone 2, this design and the resulting code, can be extended, but not modified.

Context

Recall that your client needs to have the system accommodate a range of IoT enabled coffee makers to produce coffee from orders received over the internet. The most basic coffee makers produce just the coffee of various sizes and condiments are later added manually. The more sophisticated coffee makers are programmable to also supply condiments. Your design must accommodate the ever changing range of options for coffee makers as the market evolves. Figure 1 below outlines the context for your effort.

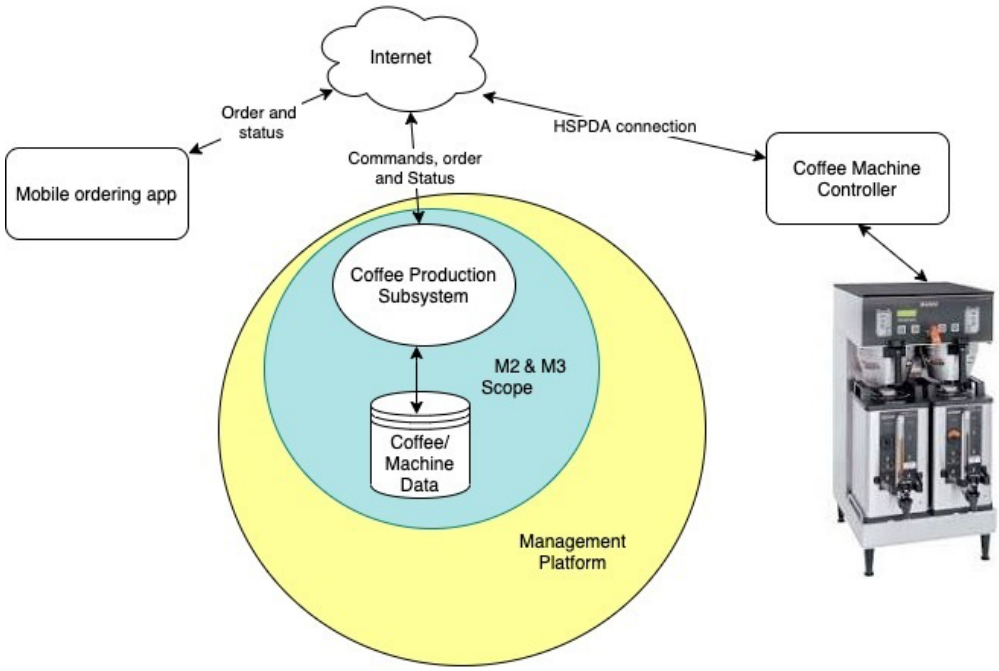

Figure 1. *The coffee production system context within the management platform.*

Note that this project milestone does not address the whole system; but rather the scope of Milestone 3 (M3) focuses your design within the CM2W Management Platform design (depicted in green). An order is sent to the Coffee Production Subsystem (CPS) via the internet from a mobile app and the coffee is either made (followed by a success message) or a message is sent to alert the system of an unsuccessful production of a coffee order and it is handled accordingly.

Scenario

With the last version of the Coffee Production Subsystem (CPS) delivered, the client is quite happy with the new capabilities to have an order for a coffee serviced via simple and automatic coffee makers, and the process managed via the CM2W Management Platform. The market has been moving to more sophisticated coffee recipes that can only be done on programmable coffee machines. These coffee makers not only can do what the simple and automated machines can do; but additionally they can handle specific instructions for the preparation of a coffee recipe which include particular amounts of ingredients (e.g., three measures of milk and two measures of French vanilla), and preparation instructions (e.g., steam milk into the cup, then add condiments, then add dark roast coffee, etc.). The assumption is that the other aspects (i.e., paying, managing inventory, managing system faults, etc.) are not in this subsystem.

For this milestone you will produce a design which builds on previous Use Cases and addresses the following new Use Cases (that are detailed along with the Data for Coffee Production in Appendix A at the end):

> **UC-3:** Order a coffee with ingredients that can be specified for a programmable coffee machine.
>
> **UC-4:** Order a coffee with specific recipes that can be sent to the programmable coffee machine, where the recipes include ingredients as well as preparation instructions.

Note that these will be implemented as part of the Coffee Production Subsystem (CPS) as depicted by the figure above. The CPS uses the Coffee/Machine Data to command the Coffee Machine(s) [via their Controller] to produce coffee that was ordered. Recall that, not all machines have the same capabilities. Simple coffee machines only produce cups of coffee and the barista or customer manually applies condiments (i.e., creamer, milk, sugar, and the like) from the counter. Automated machines can produce cups of coffee with a set of parameters for condiments (preprogrammed number of creamers, number of milks, number of sugars, and the like). Hence, with the two Use Cases described above, the focus of this milestone is to incorporate new capabilities for the more sophisticated coffee machines that can take a list of ingredients to produce the coffee, and even more take a recipe where the ingredients can have step by step instructions based on styles of coffee (e.g., cappuccino with steamed milk and the like) Note .

As this is a functional release of the product, the client would like a document containing the Software Architecture and Design, as well as a software package that can be put zipped up for the company's staff to checkout.

Learning Objectives addressed in this Milestone include:

1. Articulate the design of software using contemporary modeling techniques to reason relevant tradeoffs.
2. Apply object-oriented design principles in the creation of software systems.
3. Identify and apply appropriate object-oriented design patterns to given software problems.
4. Design and develop software systems that are easy to maintain, test, and evolve.

Milestone 1 Due Date(s)

Part 1: By **11:59 p.m., Thursday, 2/4/2021**

Initial submission must include items (at least) to the D level.

Part 2: By 11:59 p.m., Thursday, 2/11/2021; Final submission for grade.

Complete submission of all components of the assignment for Milestone 3 grade.

Grading Rubric (Basic Tasks)

F

- □ Primary Functionality not addressed.
- □ Designs are not functional/not substantiated/nonexistent.

D

- □ A defensible/discernable UML design (both Design Class Diagrams and Sequence Diagrams) is produced depicting the design of UC-3 and UC-4.
- □ The design uses both the Decorator and Factory Design Patterns.
- □ Assemble major classes into the 3 Principle Layers – both in diagrams and code.
- □ Implement and demonstrate UC-3.

C

- □ Assumptions and rationale accompany the diagrams along with descriptions of salient features and tradeoffs.
- □ UML syntax and semantics are correct in the Design Models.
- □ Design is implemented using the Decorator and Factory Design Patterns, and these are annotated in the UML as well as the code.
- □ Implement and demonstrate UC-4.

B

- □ Decorator Design Pattern is fully and correctly implemented and verified with relevant Test Cases.
- □ Implementation of the design follows the SOLID and PLK to a reasonable extent to achieve flexible code for future changes.
- □ The Software Architecture and Design is complete and presented in a consumable document with diagrams and accompanying text.

A

- □ The Factory Design Patterns are fully and correctly implemented and verified with relevant Test Cases.
- □ The UML designs should be implemented without changes from Milestone 2.
- □ The code for UC-4 should be implemented without changes from Part 1 (UC-3).
- □ The design and code are open for extension and closed for modification, as well as exhibiting other relevant SOLID and PLK principles from class.

Submitting Your Work

Please submit your Milestone 3 Part 1 document to the Moodle Milestone 3 Part 1 Assignment Dropbox; and similarly Part 2 to the Moodle Milestone 3 Part 2 Assignment

Dropbox. For both Part 1 and Part 2, please submit your document as a **pdf** file with a cover page containing your Team Name, List of team members, Assignment Title, and Date. Also include a zipfile of the project that can be imported into Eclipse, and can be built and tested (please supply a link to the repository for the source code that will be demonstrated, reviewed, and tested). Please name the document: <Team#>*Milestone3-<Part#>-DetailDesign.pdf (e.g., Team33-Milestone3-Part1-DetailedDesign.pdf)*. Please name the zipfile: <Team#>*Milestone3-<Part#>-Codebase.zip (e.g., Team33-Milestone3-Part2-Codebase.zip)*.

Appendix A:

Coffee makers have particular capabilities and the ability to make particular drinks. They are connected to a particular controller. Drinks may have particular ingredients and options. At this point, condiments (cream/sugar/flavors/etc.) are passed through to the controller rather than the machine automatically adding them.

Sample Database Instance Updates from Milestone 2:

Condiment	
Name	**Description**
Cream	Ounce serving
Sugar	Teaspoon
NutraSweet	Teaspoon
Hazelnut	Tablespoon (squirt)

Capability	
CapabilityName	**Description**
Simple	Produces drink only without any options or condiments
Automated	Produces drink with options including condiments
Programmable	Allows detailed control over order, type and amount of ingredients

CoffeeMaker		
MachineID	***Controller***	**Description**
1	1	Automated Italian
2	1 and 2	Manual expresso
3	1 and 2	Programmable

Controller			
ControllerID	**Type**	**Street_Address**	**ZIP_code**
1	Simple	200 N. Main	47803
1	Automated	200 N. Main	47803
1	Programmable	200 N. Main	47803
2	Simple	3 S. Walnut	60601
2	Programmable	3 S. Walnut	60601

DrinkTypes		
DrinkName	**Description**	**Recipe Needed**
Americano	Regular caffeinated coffee	
Latte	Coffee drink with milk and whipped cream	1. Steam milk 2. Add expresso 3. Top with whipped cream
Decaff	Regular decaffeinated coffee	
Expresso	Coffee concentrated	
Colombia Dark	Stronger roast with Colombian beans	
Pumpkin Spice	Seasonal drink w/Pumpkin	1. Add coffee 2. Add Pumpkin Spice 3. Add cream 4. Mix 5. Top with nutmeg

Ingredient	
Name	**Description**
Coffee	Reg. Coffee beans, ground
Milk	Regular dairy milk
Soy Milk	Milk-like but from soybeans
Sugar	Regular sugar cane sugar
Decaff Cofee	Reg. Decaffeinated coffee beans, ground
Whipped cream	Dairy-based heavy whipped cream
Hazelnut	Syrup
Pumpkin Spice	Spice additive
Nutmeg	Spice additive

Use Cases and Requirements

This subsystem takes an incoming order along with options and desired location and communicates with the controller co-located with the coffee machine that can make the drink. The order is assumed to be valid and the recipes for programmable coffee drinks are constructable from the information in the database. The customer only specifies the drink, options and desired location for it to be made. We're assuming there's no scheduling and that if the drink order is received, it should be made as soon as possible.

> *Also, there are some supplied interface examples (.json files) supplied in the zipfile along with this assignment that should serve as exemplars for how to communicate with the Mobile App via the order and with the Controller via the command stream and controller response.*

UC-3: Order a coffee with ingredients that can be specified for a programmable coffee machine.

Precondition(s): Orders sent from Mobile App contain only valid selections from online menu, which have ingredients indicated in the database for the coffee ordered.

Actors: Customer, controller

Steps:

1. Order, options, and desired location are submitted to system.
2. System, determines ingredients and formats the list of ingredients for a programmable coffee machine.
3. System sends order with specific ingredients to be mixed to identified controller.
4. Controller replies with status (prepared/not prepared).
5. System replies to customer with status and the coffee machine where to pick up coffee.
6. Customer picks up order at location.

Alternative Flow (no available machine that can process order or order is invalid)

Step 2: Replies to customer that order cannot be filled. Return to Step 1 to wait for orders.

Exception Flow (controller never responds to order in Step 3)

Step 4: Notifies customer that order has not been placed. Return to Step 1 to wait for orders.

Postcondition(s): Customer alerted where to pick up coffee after successful fulfillment of the order, or customer alerted to status from alternative flows.

UC-4: Order a coffee with specific recipes that can be sent to the programmable coffee machine, where the recipes include ingredients as well as preparation instructions.

Precondition(s): Orders sent from Mobile App contain only valid selections from online menu, which have predetermined recipes indicated in the database for the coffee ordered.

Actors: Customer, controller

Steps:

1. Order, options, and desired location are submitted to system.
2. System determines ingredients and preparation instructions (the recipe) for a programmable coffee machine from the order. [note: the order
3. System sends order with coffee preparation instructions (recipe) to identified controller.
4. Controller replies with status (prepared/not prepared).
5. System replies to customer with status and the coffee machine where to pick up coffee.
6. Customer picks up order at location.

Alternative Flow (no available machine that can process order or order is invalid)

Step 2: Replies to customer that order cannot be filled. Return to Step 1 to wait for orders.

Exception Flow (controller never responds to order in Step 3)

Step 4: Notifies customer that order has not been placed. Return to Step 1 to wait for orders.

Postcondition(s): Customer alerted where to pick up coffee after successful fulfillment of the order, or customer alerted to status from alternative flows.